

Multiple Kernel Sparse Representations for Supervised and Unsupervised Learning

Jayaraman J. Thiagarajan, Karthikeyan Natesan Ramamurthy and Andreas Spanias

Abstract—The success of sparse representations in image modeling has motivated its use in computer vision applications. In complex visual recognition tasks it is typical to adopt multiple descriptors, that describe different aspects of the data, for obtaining improved recognition performance. Descriptors that have diverse forms can be fused into a unified feature space in a principled manner using kernel methods. Learning sparse models in the resulting space will provide highly discriminative sparse codes for object recognition and unsupervised clustering. To this end, we develop the paradigm of multiple kernel sparse coding and propose two different approaches to optimize dictionaries for the feature space representations. The first approach works by building a separate dictionary for each descriptor set in its own feature space and then optimizes them for efficient representation in the unified feature space. Whereas, the second approach learns dictionaries in the unified feature space directly using the ensemble kernel matrices and hence provides a greater flexibility in the choice of kernel functions. Finally, we evaluate the utility of multiple kernel sparse codes obtained with the proposed approaches in object recognition and clustering applications. We demonstrate that improvements in performance are obtained by fusing multiple descriptors, when compared to using each descriptor individually.

Index Terms—Sparse coding, dictionary learning, multiple kernel learning, object recognition, clustering.

I. INTRODUCTION

A. Sparsity in Image Modeling

IMAGE understanding has been playing an increasingly crucial role in vision applications. Sparse models form an important component in image understanding, since the statistics of natural images reveal the presence of sparse structure. In [1], Olshausen and Field demonstrated that, learning sparse linear codes for natural images results in a family of features similar to those found in the primary visual cortex. Sparsity has been exploited in a variety of signal, and image processing applications including compression [2], denoising [3], compressed sensing [4], source separation [5], face classification [6], and object recognition [7]. By representing data as a sparse linear combination of atoms from a “dictionary” matrix, sparse methods lead to parsimonious models, in addition to being efficient for large scale learning. The generative model for representing a data sample $\mathbf{y} \in \mathbb{R}^M$ using the sparse code $\mathbf{a} \in \mathbb{R}^K$ can be written as

$$\mathbf{y} = \Psi \mathbf{a} + \mathbf{n}, \quad (1)$$

where Ψ is the dictionary of size $M \times K$ and \mathbf{n} is the noise component not represented using the sparse code. The goal

is to solve for the code \mathbf{a} , also known as the coefficient vector, such that it is sparse, i.e. only few of its entries differ significantly from zero. Cost functions that are commonly used to measure sparsity include the ℓ_0 and the ℓ_1 norms, which respectively measure the number of non-zero coefficients, and the sum of absolute values of the coefficients. The problem of sparse coding can be expressed as

$$\min_{\mathbf{a}} \|\mathbf{y} - \Psi \mathbf{a}\|_2 + \lambda \|\mathbf{a}\|_p, \quad (2)$$

where we set $p = 0$ or 1 to denote the ℓ_0 norm or its convex surrogate, the ℓ_1 norm, respectively, and λ is the sparsity penalty. Some of the algorithms used to solve the sparse coding problem in (2) include Basis Pursuit [8], Feature-Sign Search [9], Least Angle Regression [10], Matching Pursuit, and Orthogonal Matching Pursuit [11]. The dictionary Ψ can be predefined or obtained as an overcomplete set of vectors optimized to the training data [12]. The joint optimization problem of dictionary learning and sparse coding can be expressed as

$$\min_{\Psi, \mathbf{A}} \|\mathbf{Y} - \Psi \mathbf{A}\|_F^2 + \lambda \sum_{i=1}^N \|\mathbf{a}_i\|_p \text{ s.t. } \forall k, \|\psi_k\|_2 \leq 1, \quad (3)$$

where the training data matrix $\mathbf{Y} = [\mathbf{y}_1, \dots, \mathbf{y}_N]$, and the coefficient matrix $\mathbf{A} = [\mathbf{a}_1, \dots, \mathbf{a}_N]$. Note that (3) is not convex, and hence the sparse codes and the dictionary elements are computed using an alternating optimization procedure [2]. Furthermore, the problem of learning the dictionary has been shown to be a generalization of 1-D subspace clustering [13].

B. Sparse Coding in Object Recognition and Clustering

Sparse models have been effective in several image recovery applications, and this has motivated their use in computer vision problems. One of the first sparse coding based object recognition frameworks used codes obtained from raw image patches [14]. However, since then several frameworks have been proposed that use sparse codes of local descriptors, such as the Scale Invariant Feature Transform (SIFT), extracted from images. In order to construct image level descriptors, the codes are aggregated in an orderless bag-of-features approach [15] or using the spatial pyramid matching (SPM) approach [16] that partially preserves the spatial ordering. Methods that use sparse codes of local descriptors in spatial pyramids have achieved better performance [7], [17]–[19], in comparison to the original SPM approach which is based on vector quantization. Furthermore, the authors in [17] demonstrated that spatial pyramid aggregation of sparse codes can lead to high object recognition accuracies with just linear classifiers.

The authors are with the SenSIP Center, School of ECEE, Arizona State University, USA. 85287-5706

Email: {jjayaram, knatesan, spanias}@asu.edu

By incorporating discriminatory constraints, sparse methods can be made more effective in recognition tasks. For this reason, some algorithms explicitly incorporate class-specific discriminatory information when learning the dictionary, and this is applied for digit recognition and image classification [18], [20]–[22]. Furthermore, improved discrimination among classes have been obtained by performing simultaneous sparse coding to enforce similar non-zero coefficient support for all samples in a class [23]–[25], and coding the descriptors using dictionary atoms in their neighborhood [7]. Other approaches that lead to discriminative sparse codes are those that require the codes to obey constraints induced by the neighborhood graphs of the training data [26], [27].

In addition to their widespread applicability in supervised learning frameworks, sparsity has also been shown to be useful in unsupervised clustering applications. In [28], the authors show that clustering graph-regularized sparse codes with K-means results in better clustering performance when compared to using the data directly. The relation between data samples can be inferred by representing each data sample as a sparse linear combination of all the others. These sparse codes can be then used to build an ℓ_1 graph, and spectral clustering can be performed on the graph [29]. However, if there are a large number of training examples, obtaining sparse codes in this fashion can be computationally expensive. In [30], the authors show that this can be avoided by using a fixed size dictionary with appropriate constraints. This dictionary can be then used to obtain sparse codes and they can be subsequently used to perform spectral clustering.

C. Kernel Sparse Models

Despite its great applicability, the use of sparse models in object recognition presents two main challenges. Firstly, no single descriptor can efficiently represent the various aspects of the images. Hence, there is a need to integrate multiple descriptors extracted from images into the sparse coding paradigm. The other challenge is that each descriptor could potentially belong to a different space and the similarity measure for each descriptor could be defined as a non-linear function. The proper way to combine them hence would be to fuse the information that each descriptor provides about the images, and not the raw descriptors themselves. This can be efficiently performed by using appropriate kernel functions, which measures the similarity (possibly non-linear) between each set of descriptors [31], and combining the kernel similarities to learn sparse models in the unified feature space. The advantage of using multiple diverse features in object recognition has been demonstrated in a number of research efforts [32]–[34].

The sparse models learned in the unified feature space will lead to discriminatory codes that can be used with linear classifiers. This is because, in this space, the similarity measure between the features is linear and hence those that belong to the same class will be grouped together in subspaces. Greedy approaches to obtain sparse codes using the kernel trick have been proposed [35], [36]. In [37], Guo *et al.* proposed to perform kernel sparse coding of image descriptors, and

design dictionaries for object recognition, when the Radial Basis Function (RBF) kernel is used. Furthermore, the authors of [38] designed a kernel dictionary learning algorithm for digit recognition and demonstrated improved discrimination, particularly in the presence of noise.

D. Contributions

In this paper, we propose two different approaches for obtaining sparse codes and optimizing dictionaries in the unified feature space obtained using multiple kernels. Both the proposed methods require the extraction of appropriate features, and computation of their corresponding base kernels. The first approach, described in Section IV, works by building separate dictionaries to sparsely code each descriptor set, and fusing the corresponding kernel matrices to obtain multiple kernel sparse representations (MKSRs). The MKSR is a single sparse code that represents each training example in the unified feature space. Though the fused kernel matrices are used for evaluating MKSRs, each dictionary is optimized separately using a fixed point algorithm. In the second approach for obtaining MKSRs, we directly perform kernel dictionary learning using the ensemble kernel matrix of the training images, constructed by fusing the kernel similarities of all descriptors (Section V). In order to obtain kernel dictionaries, we propose the kernel multilevel dictionary learning (KMLD) algorithm and design a greedy procedure to obtain sparse codes using the learned dictionary. In order to evaluate the object recognition performance, we extract 8 different image descriptors, and compute MKSRs using the two proposed approaches (Section VI-A). We report the classification results on the Caltech-101 [39] and Caltech-256 [40] datasets, and present comparisons with other recent sparse coding-based recognition frameworks. Simulation results show that the proposed algorithms outperform other baseline sparse coding based object recognition approaches (Section VI-B). Finally, we demonstrate the utility of MKSRs in unsupervised learning by performing spectral clustering on the coefficient graphs. We show that using MKSRs leads to a better clustering performance, when compared to using kernel sparse codes obtained from individual features.

II. KERNEL SPARSE REPRESENTATIONS AND DICTIONARY LEARNING

The kernel function maps the non-linearly separable features into a high dimensional feature space, in which similar features are grouped together, hence improving the linear separability [31]. The authors of [37] showed that sparse representations can be efficiently performed in a high dimensional feature space using kernel methods. In this section, we review the problem of kernel sparse representations, and describe the procedure to optimize dictionaries using a fixed point iteration method when the RBF kernel is used.

Let us define a function $\Phi : \mathbb{R}^M \mapsto \mathcal{F}$, that maps the data samples from the input space to a high dimensional feature space. The data sample in the input space \mathbf{y} transforms to $\Phi(\mathbf{y})$ in the feature space and the N training examples given by $\mathbf{Y} = [\mathbf{y}_1 \dots \mathbf{y}_N]$ transform to $\Phi(\mathbf{Y}) = [\Phi(\mathbf{y}_1) \dots \Phi(\mathbf{y}_N)]$. The feature space similarity or the kernel similarity between

the training examples \mathbf{y}_i and \mathbf{y}_j is defined using the pre-defined kernel function as $\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) := \Phi(\mathbf{y}_i)^T \Phi(\mathbf{y}_j)$. The dictionary in the feature space is denoted by the matrix, $\Phi(\Psi) = [\Phi(\psi_1), \Phi(\psi_2), \dots, \Phi(\psi_K)]$, where each column indicates a dictionary element. The similarities between dictionary elements and the training examples can also be computed using the kernel function as $\Phi(\psi_k)^T \Phi(\mathbf{y}_j) = \mathcal{K}(\psi_k, \mathbf{y}_j)$ and $\Phi(\psi_k)^T \Phi(\psi_l) = \mathcal{K}(\psi_k, \psi_l)$. Since all similarities can be computed exclusively using the kernel function, it is not necessary to know the transformation Φ . This greatly simplifies the computations in the feature space when the similarities are pre-computed (*kernel trick*). We use the notation $\mathbf{K}_{\mathbf{Y}\mathbf{Y}} \in \mathbb{R}^{N \times N}$ to represent the matrix $\Phi(\mathbf{Y})^T \Phi(\mathbf{Y})$ and it contains the kernel similarities between all training examples. The similarity between two training examples ($\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j)$) is the (i, j) th element of $\mathbf{K}_{\mathbf{Y}\mathbf{Y}}$, also denoted as $\mathbf{K}_{\mathbf{y}_i \mathbf{y}_j}$.

Sparse coding can be performed in the feature space as

$$\min_{\mathbf{a}} \|\Phi(\mathbf{y}) - \Phi(\Psi)\mathbf{a}\|_2^2 + \lambda \|\mathbf{a}\|_1, \quad (4)$$

and the objective can be expanded as

$$\begin{aligned} & \Phi(\mathbf{y})^T \Phi(\mathbf{y}) - 2\mathbf{a}^T \Phi(\Psi)^T \Phi(\mathbf{y}) + \mathbf{a}^T \Phi(\Psi)^T \Phi(\Psi) \mathbf{a} + \lambda \|\mathbf{a}\|_1, \\ & = \mathbf{K}_{\mathbf{y}\mathbf{y}} - 2\mathbf{a}^T \mathbf{K}_{\Psi\mathbf{y}} + \mathbf{a}^T \mathbf{K}_{\Psi\Psi} \mathbf{a} + \lambda \|\mathbf{a}\|_1. \end{aligned} \quad (5)$$

Note that we have used the kernel trick here to simplify the computations. Following our notation, $\mathbf{K}_{\mathbf{y}\mathbf{y}}$ is the element $\mathcal{K}(\mathbf{y}, \mathbf{y})$, $\mathbf{K}_{\Psi\mathbf{y}}$ is a $K \times 1$ vector containing the elements $\mathcal{K}(\psi_k, \mathbf{y})$, for $k = \{1, \dots, K\}$ and $\mathbf{K}_{\Psi\Psi}$ is a $K \times K$ matrix containing the kernel similarities between all the dictionary atoms. Clearly, the objective function in (5) is similar to the sparse coding problem, except for the use of kernel similarities. However, the computation of kernel matrices incurs additional complexity. When the kernel sparse codes for all N training samples are computed, the dictionary can be obtained by minimizing

$$\sum_{i=1}^N \|\Phi(\mathbf{y}_i) - \Phi(\Psi)\mathbf{a}_i\|_2^2, \quad (6)$$

with respect to the constraint that they are normalized in the feature space, $\Phi(\psi_k)^T \Phi(\psi_k) = 1, \forall k = \{1, \dots, K\}$. In order to perform dictionary update in the feature space, the authors in [37] proposed a fixed point algorithm. We derive expressions for dictionary update using fixed point iteration method for the case of the RBF kernel below.

A. Dictionary Update for RBF Kernel

The radial basis function (RBF) is a well-known kernel that has wide applicability and it is defined as

$$\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) = \exp(-\gamma \|\mathbf{y}_i - \mathbf{y}_j\|_2^2), \quad (7)$$

where γ is a positive constant. Note that, $\mathcal{K}(\psi_k, \psi_k) = 1$ and hence we need not enforce the normalization constraint $\Phi(\psi_k)^T \Phi(\psi_k) = 1$ for the dictionary atoms. The objective function for dictionary update in (6) can be expressed as

$$\sum_{i=1}^N \left[1 - 2 \sum_{l=1}^K a_{l,i} \mathcal{K}(\psi_l, \mathbf{y}_i) + \sum_{l=1}^K \sum_{t=1}^K a_{l,i} a_{t,i} \mathcal{K}(\psi_l, \psi_t) \right],$$

where $a_{l,i}$ represents the i^{th} element in the coefficient vector \mathbf{a}_l . In order to update the dictionary atom ψ_k , we compute the derivative of the simplified objective using the definition of RBF kernel in (7), and set the derivative to zero as follows,

$$\begin{aligned} & -4\gamma \sum_{i=1}^N \left[-a_{k,i} \mathcal{K}(\psi_k, \mathbf{y}_i) (\psi_k - \mathbf{y}_i) \right. \\ & \quad \left. + \sum_{t=1}^K a_{k,i} a_{t,i} \mathcal{K}(\psi_k, \psi_t) (\psi_k - \psi_t) \right] = 0. \end{aligned} \quad (8)$$

To update the dictionary atoms we can employ a fixed point procedure, where the dictionary atom ψ_k from the $(n-1)^{\text{th}}$ is used to compute the kernel similarities for the n^{th} iteration of the update. Denoting the k^{th} atom in the n^{th} iteration by $\psi_k^{(n)}$, we can rewrite the expression in (8) as

$$\begin{aligned} & -4\gamma \sum_{i=1}^N \left[-a_{k,i} \mathcal{K}(\psi_k^{(n-1)}, \mathbf{y}_i) (\psi_k^{(n)} - \mathbf{y}_i) \right. \\ & \quad \left. + \sum_{t=1}^K a_{k,i} a_{t,i} \mathcal{K}(\psi_k^{(n-1)}, \psi_t) (\psi_k^{(n)} - \psi_t) \right] = 0. \end{aligned} \quad (9)$$

Solving (9), we obtain

$$\psi_k^{(n)} = \frac{\Psi \text{diag}[\mathbf{K}_{\psi_k^{(n-1)} \Psi}] \mathbf{A} \mathbf{a}_{k, \text{row}}^T - \mathbf{Y} \text{diag}[\mathbf{K}_{\psi_k^{(n-1)} \mathbf{Y}}] \mathbf{a}_{k, \text{row}}^T}{\mathbf{K}_{\psi_k^{(n-1)} \Psi} \mathbf{A} \mathbf{a}_{k, \text{row}}^T - \mathbf{K}_{\psi_k^{(n-1)} \mathbf{Y}} \mathbf{a}_{k, \text{row}}^T}.$$

Here $\mathbf{a}_{k, \text{row}}$ is a row vector containing the set of coefficients of all training vectors corresponding to the dictionary atom ψ_k , and $\text{diag}[\cdot]$ creates a diagonal matrix using the argument vector as its diagonal. The $1 \times K$ kernel matrix $\mathbf{K}_{\psi_k^{(n-1)} \Psi}$ contains the elements $\mathcal{K}(\psi_k^{(n-1)}, \psi_l)$, for $l = \{1 \dots K\}$ and the $1 \times N$ matrix $\mathbf{K}_{\psi_k^{(n-1)} \mathbf{Y}}$ contains the elements $\mathcal{K}(\psi_k^{(n-1)}, \mathbf{y}_i)$, for $i = \{1 \dots N\}$ respectively.

III. MULTIPLE KERNEL SPARSE REPRESENTATIONS

The use of multiple descriptors to characterize images has been a very successful approach for complex visual recognition tasks. Though this method provides the flexibility of choosing features to describe different aspects of the underlying data, the resulting representations are high-dimensional, and the descriptors can be very diverse. Hence, in order to facilitate recognition tasks, there is a need to transform these features to a unified space, and construct low dimensional compact representations for the images in the unified space. Multiple Kernel Learning (MKL) provides a convenient way of fusing multiple descriptors by combining multiple base kernels, each of which is created based on an image descriptor [32]. In this work, we develop the Multiple Kernel Sparse Representations (MKSR) model that aims to compute the sparse representation for an image in the unified feature space obtained using multiple kernels. In addition to building a compact representation for the data in the high-dimensional feature space, MKSR can provide highly discriminative codes.

Let us denote the set of descriptors obtained from the data sample as $\mathbf{y}_i = \{\mathbf{y}_{i,r}\}_{r=1}^R$, where i is the sample index, r is the descriptor index, and R is the total number of descriptors. Let the set of R base kernel functions, each

corresponding to a descriptor be denoted as $\{\mathcal{K}_r\}_{r=1}^R$, and the base kernel matrices be denoted as $\{\mathbf{K}_r\}_{r=1}^R$. The ensemble kernel function and matrix can be constructed as the non-negative linear combination,

$$\mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) = \sum_{r=1}^R \beta_r \mathcal{K}_r(\mathbf{y}_{i,r}, \mathbf{y}_{j,r}), \quad \forall \beta_r \geq 0, \quad (10)$$

$$\mathbf{K} = \sum_{r=1}^R \beta_r \mathbf{K}_r, \quad \forall \beta_r \geq 0. \quad (11)$$

As described earlier, the ensemble kernel matrix contains the similarities between the data samples in the unified feature space obtained using multiple kernels. Various types of descriptors such as raw pixel values, histograms, feature vectors, and spatial pyramids can be successfully combined by considering only the kernel matrices corresponding to the descriptors. Given a distance function ρ_r measuring the distance between the r^{th} descriptors of two samples, we construct the kernel matrix as

$$\mathbf{K}_r(i, j) = \mathcal{K}_r(\mathbf{y}_{i,r}, \mathbf{y}_{j,r}) = \exp(-\gamma \rho_r^2(\mathbf{y}_{i,r}, \mathbf{y}_{j,r})), \quad (12)$$

where γ is a positive constant. Computing this kernel function is convenient, since several useful image descriptors and their corresponding distance functions have been proposed in the literature. Note that the kernel matrix in (12) is not guaranteed to be positive semidefinite, which is a required characteristic for a proper kernel matrix. In such cases, we follow the approach in [41] where we compute the smallest eigenvalue of \mathbf{K}_r and if it is negative, we add its absolute value to the diagonal of \mathbf{K}_r .

Given a dictionary Ψ and a kernel function \mathcal{K} , sparse codes can be obtained as described in Section II. However, in the MKSR model we have R different descriptors for each image. In order to obtain sparse codes using an ensemble kernel matrix, we need to optimize the dictionaries for the unified feature space. In this paper, we consider two different approaches for obtaining multiple kernel sparse codes and designing dictionaries using the kernel trick and they will be described in Sections IV and V respectively. The first approach computes separate dictionaries for each descriptor, and obtains sparse codes in the unified feature space. The second approach directly builds a multilevel dictionary [42] using the ensemble kernel matrix in the unified feature space. As a result, this approach does not require knowledge of the mathematical form of the individual kernels. On the other hand, the advantage with the first approach is that optimized dictionaries for each descriptor, if previously available, can be readily used to initialize the learning algorithm.

IV. PROPOSED METHOD 1

In this section, we present the *Method 1*, that alternatively optimizes separate dictionaries for each image descriptor and obtains MKSRs by fusing the individual kernels. Consider a dataset of N data samples and R different descriptors that characterize them. The kernel function \mathcal{K}_r for descriptor r can be pre-defined kernel or constructed using the distance function d_r . For the r^{th} descriptor, we use its corresponding samples, $\{\mathbf{y}_{i,r}\}_{i=1}^N$, to learn the dictionary $\Psi_r =$

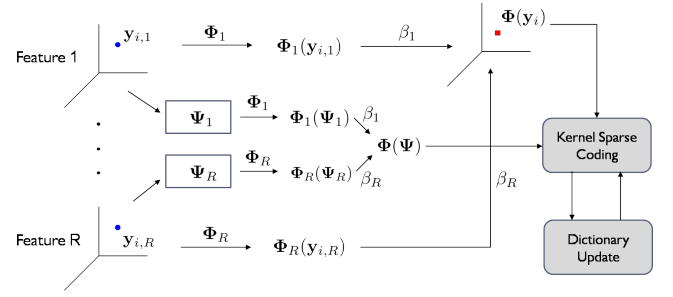


Fig. 1. Proposed *Method 1* for obtaining multiple kernel sparse representations. In this approach, we alternatively optimize the individual dictionaries $\{\Psi_r\}_{r=1}^R$ and obtain the unified sparse codes $\{\mathbf{a}_i\}_{i=1}^N$.

$[\psi_{1,r}, \psi_{2,r}, \dots, \psi_{K,r}]$ that can sparsely represent the descriptor set in its feature space. We compute the kernel matrices as $\mathbf{K}_{\mathbf{Y}\mathbf{Y},r}(i, j) = \mathcal{K}_r(\mathbf{y}_{i,r}, \mathbf{y}_{j,r})$, $\mathbf{K}_{\Psi\mathbf{Y},r}(k, i) = \mathcal{K}_r(\psi_{k,r}, \mathbf{y}_{i,r})$, and $\mathbf{K}_{\Psi\Psi,r}(k, l) = \mathcal{K}_r(\psi_{k,r}, \psi_{l,r})$ respectively. Using the R descriptors, we obtain the ensemble kernel matrices as,

$$\mathbf{K}_{\mathbf{Y}\mathbf{Y}}(i, j) = \sum_{r=1}^R \beta_r \mathbf{K}_{\mathbf{Y}\mathbf{Y},r}(i, j), \quad \forall \beta_r \geq 0, \quad (13)$$

$$\mathbf{K}_{\Psi\mathbf{Y}}(k, i) = \sum_{r=1}^R \beta_r \mathbf{K}_{\Psi\mathbf{Y},r}(k, i), \quad \forall \beta_r \geq 0, \quad (14)$$

$$\mathbf{K}_{\Psi\Psi}(k, l) = \sum_{r=1}^R \beta_r \mathbf{K}_{\Psi\Psi,r}(k, l), \quad \forall \beta_r \geq 0. \quad (15)$$

The objective function to be minimized for MKSR can be expressed as

$$\begin{aligned} \min_{\mathbf{a}} & \|\Phi(\mathbf{y}) - \Phi(\Psi)\mathbf{a}\|_2^2 + \lambda \|\mathbf{a}\|_1, \\ & = \mathbf{K}_{\mathbf{Y}\mathbf{Y}} - 2\mathbf{a}^T \mathbf{K}_{\Psi\mathbf{Y}} + \mathbf{a}^T \mathbf{K}_{\Psi\Psi} \mathbf{a} + \lambda \|\mathbf{a}\|_1. \end{aligned} \quad (16)$$

Here $\Phi(\mathbf{y})$ and $\Phi(\Psi)$ denote the data sample and the dictionary in the multiple kernel domain. The dictionaries and the MKSRs are hence computed in an alternating fashion until the objective is minimized.

A. Updating the Dictionaries

As illustrated in Figure 1, the proposed *Method 1* fuses multiple dictionaries to perform sparse coding in the unified feature space. Optimization of the dictionaries for efficient coding needs to be carried out using a fixed point algorithm as described in Section II. Rewriting the objective function for dictionary update, $\sum_{i=1}^N \|\Phi(\mathbf{y}_i) - \Phi(\Psi)\mathbf{a}_i\|_2^2$, as

$$\begin{aligned} & = \sum_{i=1}^N \left[\mathcal{K}(\mathbf{y}_i, \mathbf{y}_i) - 2 \sum_{l=1}^K a_{l,i} \mathcal{K}(\psi_l, \mathbf{y}_i) \right. \\ & \quad \left. + \sum_{l=1}^K \sum_{t=1}^K a_{l,i} a_{t,i} \mathcal{K}(\psi_l, \psi_t) \right], \\ & = \sum_{i=1}^N \left[\sum_{r=1}^R \beta_r \mathcal{K}_r(\mathbf{y}_{i,r}, \mathbf{y}_{i,r}) - 2 \sum_{l=1}^K a_{l,i} \sum_{r=1}^R \beta_r \mathcal{K}_r(\psi_{l,r}, \mathbf{y}_{i,r}) \right. \\ & \quad \left. + \sum_{l=1}^K \sum_{t=1}^K a_{l,i} a_{t,i} \sum_{r=1}^R \beta_r \mathcal{K}_r(\psi_{l,r}, \psi_{t,r}) \right]. \end{aligned} \quad (17)$$

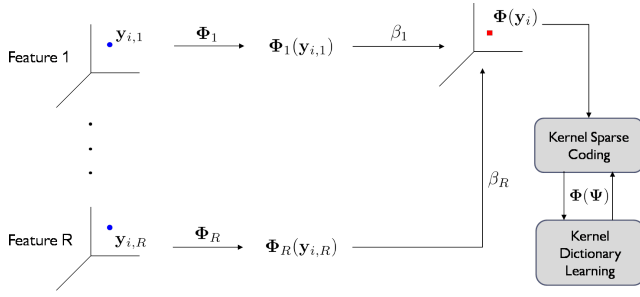


Fig. 2. Proposed *Method 2* for obtaining multiple kernel sparse representations. In this approach, we perform dictionary learning using the ensemble kernel in the unified feature space directly.

In (17), we have expanded the ensemble kernel function in terms of its individual base kernels. Since there are R different dictionaries with K atoms each, the fixed point algorithm needs to update the atoms of one dictionary at a time, fixing the other $R - 1$ dictionaries. Hence, to update the k^{th} atom of the Ψ_r , we compute the derivative of (17) with respect to $\psi_{k,r}$. Excluding all terms in (17) that do not depend on the current dictionary atom being updated, the objective can be simplified as

$$= \sum_{i=1}^N \left[-2 \sum_{l=1}^K a_{l,i} \beta_r \mathcal{K}_r(\psi_{l,r}, y_{i,r}) + \sum_{l=1}^K \sum_{t=1}^K a_{l,i} a_{t,i} \beta_r \mathcal{K}_r(\psi_{l,r}, \psi_{t,r}) \right]. \quad (18)$$

The objective in (18) is equivalent to updating the dictionary atoms for kernel sparse coding, with the kernel function \mathcal{K}_r . Note that the same set of coefficients $\{a_i\}_{i=1}^N$, are used for updating all the R dictionaries, since the representation is computed in the unified feature space. In order to obtain dictionary update expressions as shown in Section II, the kernel function should be differentiable. Furthermore, the ensemble kernel weights $\{\beta_r\}_{r=1}^R$ balance the relative importance of each feature and hence they are chosen empirically such that best classification performance is obtained.

B. Computing Sparse Codes for Test Data

Given the test data sample \mathbf{x} , we extract the R different descriptors $\{\mathbf{x}_r\}_{r=1}^R$ and evaluate the ensemble kernel matrix $\mathbf{K}_{\Psi\mathbf{x}}$ of size $K \times 1$ where

$$\mathbf{K}_{\Psi\mathbf{x}}(k) = \sum_{r=1}^R \beta_r \mathcal{K}_r(\psi_{k,r}, \mathbf{x}_r). \quad (19)$$

The kernel weights $\{\beta_r\}_{r=1}^R$, evaluated empirically during the training stage, are used with the test data. The sparse code \mathbf{b} is then obtained by minimizing the objective

$$-2\mathbf{b}^T \mathbf{K}_{\Psi\mathbf{x}} + \mathbf{b}^T \mathbf{K}_{\Psi\Psi} \mathbf{b} + \lambda \|\mathbf{b}\|_1. \quad (20)$$

V. PROPOSED METHOD 2

An alternative approach to optimizing the dictionary for sparse coding in the unified feature space is to perform dictionary learning directly using the ensemble kernel matrix

of descriptors obtained from the training data. We learn a hierarchical dictionary in multiple levels and refer to this as kernel multilevel dictionary (KMLD) learning. This algorithm is a generalization of the MLD algorithm proposed in [43] to the feature space. Such an approach eliminates the need to explicitly optimize separate dictionaries and hence provides the flexibility to choose any kernel function. The proposed framework, referred to as *Method 2*, is illustrated in Figure 2. In [43], it is shown that global dictionaries, obtained by performing 1-D subspace clustering in multiple levels, generalize better to novel test samples when compared to other dictionary learning methods. In this section, we begin by briefly discussing K-lines clustering, the 1-D subspace clustering procedure, and the multilevel dictionary (MLD) learning algorithm. We then present the kernelized version of the K-lines clustering algorithm [44], and develop the kernel MLD learning algorithm for designing dictionaries using multiple kernels. Finally, we describe the procedure to compute the sparse code for a test sample using a kernel MLD.

A. K-lines Clustering Algorithm

K-lines clustering algorithm is an iterative procedure that performs a least squares fit of K 1-D subspaces to the training data [45]. Given the set of T data samples $\mathbf{Y} = \{\mathbf{y}_i\}_{i=1}^N$ and the number of clusters K , K-lines clustering proceeds in two stages after initialization: the cluster assignment and the cluster centroid update. In the cluster assignment stage, a training vector \mathbf{y}_i is assigned to a cluster k based on the following distortion measure.

$$d(\mathbf{y}_i, \psi_k) = \|\mathbf{y}_i - \psi_k(\mathbf{y}_i^T \psi_k)\|_2^2. \quad (21)$$

Cluster assignment results in update of the K membership sets, $\{\mathcal{C}_k\}_{k=1}^K$. In the cluster centroid update stage, given the set \mathcal{C}_k , the k^{th} cluster centroid is updated as

$$\min_{\psi_k} \sum_{i \in \mathcal{C}_k} \|\mathbf{y}_i - \psi_k(\mathbf{y}_i^T \psi_k)\|_2^2 \text{ subj. to } \|\psi_k\|_2^2 = 1. \quad (22)$$

The principal left singular vector from the singular value decomposition (SVD) of $\mathbf{Y}_k = [\mathbf{y}_i]_{i \in \mathcal{C}_k}$ is the centroid of cluster k . Note that, we compute the principal left singular vector using an iterative procedure, similar to the Power method. We rewrite (22) as

$$\min_{\psi_k, \{a_{k,i}\}} \sum_{i \in \mathcal{C}_k} \|\mathbf{y}_i - a_{k,i} \psi_k\|_2^2 \text{ subj. to } \|\psi_k\|_2^2 = 1, \quad (23)$$

where $\{a_{k,i}\}$ is the set of coefficients for the data samples with $i \in \mathcal{C}_k$, and solve alternatively for ψ_k and $\{a_{k,i}\}$. Fixing ψ_k , we can compute $\{a_{k,i}\}$ as

$$a_{k,i} = \mathbf{y}_i^T \psi_k, \forall i \in \mathcal{C}_k. \quad (24)$$

Incorporating the constraint $\|\psi_k\|_2 = 1$ and assuming $\{a_{k,i}\}$ to be known, we can compute the cluster center as

$$\psi_k = \frac{\sum_{i \in \mathcal{C}_k} a_{k,i} \mathbf{y}_i}{\left\| \sum_{i \in \mathcal{C}_k} a_{k,i} \mathbf{y}_i \right\|_2}. \quad (25)$$

The centroid and the coefficients can be obtained for each cluster by repeating (24) and (25) for a sufficient number of iterations.

TABLE I
THE KERNEL K-LINES CLUSTERING ALGORITHM.

Input
$\mathbf{Y} = [\mathbf{y}_i]_{i=1}^N$, $M \times N$ matrix of data samples.
$\mathbf{K}_{\mathbf{Y}\mathbf{Y}}$, $N \times N$ kernel matrix.
K , desired number of clusters.
Initialization
- Randomly group data samples to initialize the membership matrix \mathbf{Z} .
- Based on \mathbf{Z} , obtain the rank-1 SVD for each cluster to initialize Ψ .
- Compute the initial coefficients, $\mathbf{A} = \mathbf{Y}^T \Psi$.
Algorithm
Loop until convergence
Loop for L iterations
- Compute $\mathbf{H} = \mathbf{Z} \odot \mathbf{A}$.
- Compute $\mathbf{A} = \mathbf{K}_{\mathbf{Y}\mathbf{Y}} \mathbf{H} \Gamma(\Phi(\mathbf{Y}) \mathbf{H})^{-1}$.
end
- Update \mathbf{Z} by identifying the index of absolute maximum in each row of \mathbf{A} .
end

To express this procedure using matrices, we define the membership matrix $\mathbf{Z} \in \mathbb{R}^{N \times K}$, where $z_{ik} = 1$ if and only if $i \in \mathcal{C}_k$. Cluster assignment is performed by computing

$$\mathbf{A} = \mathbf{Y}^T \Psi, \quad (26)$$

and then setting $\mathbf{Z} = g(\mathbf{A})$, where $g(\cdot)$ is a function that operates on a matrix and returns 1 at the location of absolute maximum of each row and zero elsewhere. Let us also define the matrix $\mathbf{H} = \mathbf{Z} \odot \mathbf{A}$, where \odot indicates the Hadamard product. The centroid update can be then performed as

$$\Psi = \mathbf{Y} \mathbf{H} \Gamma(\mathbf{Y} \mathbf{H})^{-1}, \quad (27)$$

which is the matrix version of the update equation given in (25). Here, $\Gamma(\cdot)$ is a function that operates on a matrix and returns a diagonal matrix with the ℓ_2 norm of each column in the argument matrix as its diagonal element. In (27), $\Gamma(\mathbf{Y} \mathbf{H})^{-1}$ ensures that the columns of Ψ are normalized. K-lines clustering is hence performed by iterating over membership update and centroid computation steps.

B. Multilevel Dictionary Learning

The MLD learning algorithm proposed in [43] uses a hierarchical approach by employing K-lines clustering to adapt atoms in each level of the dictionary. We denote the multilevel dictionary as $\Psi = [\Psi_1 \Psi_2 \dots \Psi_S]$, and the coefficient matrix as $\mathbf{A} = [\mathbf{A}_1^T \mathbf{A}_2^T \dots \mathbf{A}_S^T]^T$. Here, Ψ_s is the sub-dictionary in level s and \mathbf{A}_s is corresponding the coefficient matrix in level s . The approximation in level s is expressed as

$$\mathbf{R}_{s-1} = \Psi_s \mathbf{A}_s + \mathbf{R}_s, \text{ for } s = 1, \dots, S, \quad (28)$$

where \mathbf{R}_{s-1} , \mathbf{R}_s are the residuals for the levels $s-1$ and s respectively and $\mathbf{R}_0 = \mathbf{Y}$. This implies that the residuals in level $s-1$ serve as the training data for level s . Note that the sparsity of the representation in each level is fixed at 1. K-lines clustering is employed to learn Ψ_s from the training matrix, \mathbf{R}_{s-1} , for that level. Detailed discussion on the generalization characteristics of multilevel learning can be found in [42].

C. Kernel K-lines Clustering Algorithm

In this section, we propose an algorithm to perform K-lines clustering in the feature space using kernel similarities. Transformation of data to an appropriate feature space leads to tighter clusters and hence developing a kernel version of the K-lines clustering algorithm may lead to an improved clustering accuracy. Using the transformed data samples and dictionary elements, the coefficient matrix, \mathbf{A} , in feature space can be computed in a manner similar to (26),

$$\mathbf{A} = \Phi(\mathbf{Y})^T \Phi(\Psi), \quad (29)$$

and the membership matrix is given by $\mathbf{Z} = g(\mathbf{A})$. Hence, the cluster centers in the feature space can be obtained as

$$\Phi(\Psi) = \Phi(\mathbf{Y}) \mathbf{H} \Gamma(\Phi(\mathbf{Y}) \mathbf{H})^{-1}, \quad (30)$$

where $\mathbf{H} = \mathbf{Z} \odot \mathbf{A}$. The normalization term is computed as

$$\begin{aligned} \Gamma(\Phi(\mathbf{Y}) \mathbf{H}) &= \text{diag}((\Phi(\mathbf{Y}) \mathbf{H})^T \Phi(\mathbf{Y}) \mathbf{H})^{1/2}) \\ &= \text{diag}(\mathbf{H}^T \mathbf{K}_{\mathbf{Y}\mathbf{Y}} \mathbf{H})^{1/2}, \end{aligned} \quad (31)$$

where $\text{diag}(\cdot)$ is an operator that returns a diagonal matrix with the diagonal elements same as that of the argument matrix. Combining (29), (30) and (31), we obtain

$$\mathbf{A} = \Phi(\mathbf{Y})^T \Phi(\mathbf{Y}) \mathbf{H} \Gamma(\Phi(\mathbf{Y}) \mathbf{H})^{-1} = \mathbf{K}_{\mathbf{Y}\mathbf{Y}} \mathbf{H} \Gamma(\Phi(\mathbf{Y}) \mathbf{H})^{-1}.$$

The steps of this algorithm are presented in Table I.

D. Kernel Multilevel Dictionary Learning Algorithm

Given a set of training samples, our goal is to design multilevel dictionaries in the unified feature space obtained using multiple kernels. The kernel K-lines clustering procedure developed in the previous section can be used to learn the atoms in every level of the dictionary. In level s , we denote the sub-dictionary by $\Phi(\Psi_s)$, the membership matrix by \mathbf{Z}_s , the coefficient matrix by \mathbf{A}_s , the input and the residual matrices by $\Phi(\mathbf{Y}_s)$ and $\Phi(\mathbf{R}_s)$ respectively. The training set for the first level is $\mathbf{Y}_1 = \mathbf{Y}$. Given the N training images, we build the R descriptors and compute the ensemble kernel matrix $\mathbf{K}_{\mathbf{Y}\mathbf{Y}}$ as given in (13),

$$\mathbf{K}_{\mathbf{Y}\mathbf{Y}}(i, j) = \mathcal{K}(\mathbf{y}_i, \mathbf{y}_j) = \sum_{r=1}^R \beta_r \mathcal{K}_r(\mathbf{y}_{i,r}, \mathbf{y}_{j,r}). \quad (32)$$

As described in the previous section, we can compute $\mathbf{H}_s = \mathbf{Z}_s \odot \mathbf{A}_s$. Performing kernel K-lines clustering in level 1 will yield the coefficients $\mathbf{A}_1 = \mathbf{K}_{\mathbf{Y}\mathbf{Y}} \mathbf{H}_1 \mathbf{D}_1$, where $\mathbf{D}_1 = \Gamma(\Phi(\mathbf{Y}_1) \mathbf{H}_1)^{-1} = \text{diag}(\mathbf{H}_1^T \mathbf{K}_{\mathbf{Y}\mathbf{Y}} \mathbf{H}_1)^{-1/2}$ indicates the diagonal matrix that normalizes the dictionary atoms of level 1 in the feature space. In kernel MLD learning, the residual vectors in a level are used as the training set to the next level. Hence, we compute the residuals as

$$\begin{aligned} \Phi(\mathbf{R}_1) &= \Phi(\mathbf{Y}_1) - \Phi(\Psi_1) \mathbf{H}_1^T = \Phi(\mathbf{Y}_1) - \Phi(\mathbf{Y}_1) \mathbf{H}_1 \mathbf{D}_1 \mathbf{H}_1^T, \\ &= \Phi(\mathbf{Y}_1) [\mathbf{I} - \mathbf{H}_1 \mathbf{D}_1 \mathbf{H}_1^T] = \Phi(\mathbf{Y}_2). \end{aligned} \quad (33)$$

Given the residuals from level 1, the dictionary atoms in level 2 can be computed as $\Phi(\Psi_2) = \Phi(\mathbf{Y}_2) \mathbf{H}_2 \mathbf{D}_2$, where $\mathbf{D}_2 =$

TABLE II
KERNEL MULTILEVEL DICTIONARY LEARNING ALGORITHM.

Input
$\mathbf{Y}_1 = [\mathbf{y}_{1,i}]_{i=1}^N$, $D \times N$ matrix of training image patches.
$\mathbf{K}_{\mathbf{Y}\mathbf{Y}}$, $N \times N$ kernel matrix.
K , desired number of atoms per level.
S , total number of levels.
Initialization
- Randomly initialize the membership \mathbf{Z}_1 and compute the initial coefficients, \mathbf{A}_1 .
Algorithm
For $s = 1$ to S
Loop until convergence
- Loop for L iterations
- Compute $\mathbf{H}_s = \mathbf{Z}_s \odot \mathbf{A}_s$.
- Compute $\mathbf{D}_s = \text{diag} \left[\mathbf{H}_s^T \left(\prod_{t=1}^{s-1} (\mathbf{I} - \mathbf{H}_t \mathbf{D}_t \mathbf{H}_t^T) \right) \right]^T \times \mathbf{K}_{\mathbf{Y}\mathbf{Y}} \left(\prod_{t=1}^{s-1} (\mathbf{I} - \mathbf{H}_t \mathbf{D}_t \mathbf{H}_t^T) \right) \mathbf{H}_s \right]^{-1/2}$.
- Evaluate $\mathbf{A}_s = \left[\left(\prod_{t=1}^{s-1} (\mathbf{I} - \mathbf{H}_t \mathbf{D}_t \mathbf{H}_t^T) \right)^T \mathbf{K}_{\mathbf{Y}\mathbf{Y}} \times \left(\prod_{t=1}^{s-1} (\mathbf{I} - \mathbf{H}_t \mathbf{D}_t \mathbf{H}_t^T) \right) \right] \mathbf{H}_s \mathbf{D}_s$.
end
- Update \mathbf{Z}_s using index of absolute maximum in each row of \mathbf{A}_s .
end
end

$\text{diag}((\Phi(\mathbf{Y}_2)\mathbf{H}_2)^T(\Phi(\mathbf{Y}_2)\mathbf{H}_2))^{-1/2}$. This is simplified as

$$\mathbf{D}_2 = \text{diag}[\mathbf{H}_2^T (\mathbf{I} - \mathbf{H}_1 \mathbf{D}_1 \mathbf{H}_1^T)^T \mathbf{K}_{\mathbf{Y}\mathbf{Y}} (\mathbf{I} - \mathbf{H}_1 \mathbf{D}_1 \mathbf{H}_1^T) \mathbf{H}_2]^{-1/2}. \quad (34)$$

Similar to the previous level, the coefficients are evaluated as

$$\begin{aligned} \mathbf{A}_2 &= \Phi(\mathbf{Y}_2)^T \Phi(\mathbf{Y}_2), \\ &= (\mathbf{I} - \mathbf{H}_1 \mathbf{D}_1 \mathbf{H}_1^T)^T \mathbf{K}_{\mathbf{Y}\mathbf{Y}} (\mathbf{I} - \mathbf{H}_1 \mathbf{D}_1 \mathbf{H}_1^T) \mathbf{H}_2 \mathbf{D}_2. \end{aligned} \quad (35)$$

Table II shows the detailed algorithm to learn a kernel MLD by generalizing the procedure for S levels. Note that the innermost loop in the algorithm computes the cluster centroids using the linearized SVD procedure (Section V-A). The middle loop performs the kernel K-lines clustering for a particular level. Similar to *Method 1*, the weights $\{\beta_r\}_{r=1}^R$ are tuned empirically to provide the best classification performance.

E. Computing Sparse Codes for Test Data

In this section, we describe a procedure to evaluate the sparse code for a test sample using the kernel MLD. Using the R descriptors $\{\mathbf{x}_r\}_{r=1}^R$ extracted from a test sample, we evaluate the ensemble kernel matrix $\mathbf{K}_{\mathbf{x}\mathbf{Y}} \in \mathbb{R}^{1 \times N}$ where

$$\mathbf{K}_{\mathbf{x}\mathbf{Y}}(i) = \mathcal{K}(\mathbf{x}, \mathbf{y}_i) = \sum_{r=1}^R \beta_r \mathcal{K}_r(\mathbf{x}_r, \mathbf{y}_{i,r}). \quad (36)$$

In order to obtain sparse codes for the test sample using the kernel MLD, we compute a sparse coefficient for each level

using the dictionary atoms from that level. Similar to the training procedure, we first compute the correlations between the test sample and all dictionary elements in level 1 as

$$\alpha_1 = \Phi(\mathbf{x})^T \Phi(\mathbf{Y}_1) = \Phi(\mathbf{x})^T \Phi(\mathbf{Y}_1) \mathbf{H}_1 \mathbf{D}_1 = \mathbf{K}_{\mathbf{x}\mathbf{Y}} \mathbf{H}_1 \mathbf{D}_1.$$

Following this, we determine the $1 \times K$ membership vector $\mathbf{z}_1 = g(\alpha_1)$ and the coefficient vector $\mathbf{h}_1 = \mathbf{z}_1 \odot \alpha_1$. The residual vector of the test sample can be computed as

$$\Phi(\mathbf{r}_1) = \Phi(\mathbf{x}) - \Phi(\mathbf{Y}_1) \mathbf{h}_1^T = \Phi(\mathbf{x}) - \Phi(\mathbf{Y}_1) \mathbf{H}_1 \mathbf{D}_1 \mathbf{h}_1^T.$$

To determine a 1-sparse code in level 2, the residual vector \mathbf{r}_1 needs to be correlated with the dictionary atoms $\Phi(\mathbf{Y}_2)$. Generalizing this to any level s , we can evaluate the correlations between the residual $\Phi(\mathbf{r}_{s-1})$ and the dictionary atoms $\Phi(\mathbf{Y}_s)$ as

$$\alpha_s = \mathbf{M}_s \mathbf{H}_s \mathbf{D}_s, \quad (37)$$

where \mathbf{M}_s is given by

$$\begin{aligned} &= \left[\mathbf{K}_{\mathbf{x}\mathbf{Y}} - \sum_{t=1}^{s-1} \mathbf{h}_t \mathbf{D}_t \mathbf{H}_t^T \left(\prod_{p=1}^{t-1} (\mathbf{I} - \mathbf{H}_p \mathbf{D}_p \mathbf{H}_p^T) \right)^T \mathbf{K}_{\mathbf{Y}\mathbf{Y}} \right] \\ &\quad \times \left[\left(\prod_{t=1}^{s-1} (\mathbf{I} - \mathbf{H}_t \mathbf{D}_t \mathbf{H}_t^T) \right) \right]. \end{aligned} \quad (38)$$

The coefficient vector corresponding to level s can then be obtained as $\mathbf{h}_s = \mathbf{z}_s \odot \alpha_s$, where $\mathbf{z}_s = g(\alpha_s)$. The overall sparse code for the test sample \mathbf{x} can be constructed by stacking coefficient vectors from all levels, $\mathbf{b} = [\mathbf{h}_1^T \mathbf{h}_2^T \dots \mathbf{h}_S^T]^T$. To compute the sparse code for a test sample \mathbf{x} , *Method 1* works with the ensemble kernel similarities $\mathbf{K}_{\Psi\mathbf{x}} \in \mathbb{R}^K$ and since $K < N$, it is computationally less intensive compared to *Method 2*, which uses the kernel similarities $\mathbf{K}_{\mathbf{x}\mathbf{Y}} \in \mathbb{R}^N$. However, *Method 2* has the important advantage that it does not place any restriction on the choice of the kernel function or the distance function for building the kernel matrix.

VI. OBJECT RECOGNITION AND UNSUPERVISED CLUSTERING

In this section, we describe the set of image descriptors and the kernel functions considered for our simulations and present discussions on the recognition performance using Caltech-101 and Caltech-256 datasets. The same set of features were used to compute sparse coding-based graphs for clustering. In all cases, we constructed the kernel matrices using (12), with appropriately chosen distance functions between descriptors.

A. Image Descriptors and Kernels

SIFT-ScSPM: For a given image, we extracted SIFT features [46] with three scales on a dense grid and used a K-means dictionary to obtain sparse codes for the local features. The number of dictionary elements was fixed at 1024. For each image, we generated the ScSPM feature using the algorithm in [17] and aggregated sparse codes using max-pooling at spatial scales 1, 2 and 4 respectively. The kernel matrix was constructed based on Euclidean distance between the features. **SS-ScSPM:** The base kernel was constructed in the same way

TABLE III
COMPARISON OF THE CLASSIFICATION ACCURACIES ON THE
CALTECH-101 DATASET.

Method	# Training samples per class					
	5	10	15	20	25	30
Zhang <i>et.al.</i> [41]	46.6	55.8	59.1	62	-	66.2
Lazebnik <i>et.al.</i> [16]	-	-	56.4	-	-	64.6
Griffin <i>et.al.</i> [54]	44.2	54.5	59	63.3	65.8	67.6
Jain <i>et.al.</i> [55]	-	-	65	-	-	70.4
Boiman <i>et.al.</i> [56]	-	-	61	-	-	69.1
Pham <i>et.al.</i> [57]	-	-	42	-	-	-
Gemert <i>et.al.</i> [58]	-	-	-	-	-	64.16
Yang <i>et.al.</i> [17]	-	-	67	-	-	73.2
Wang <i>et.al.</i> [19]	51.15	59.77	65.43	67.74	70.16	73.44
Aharon <i>et.al.</i> [2]	49.8	59.8	65.2	68.7	71	73.2
Zhang <i>et.al.</i> [22]	49.6	59.5	65.1	68.6	71.1	73
Jiang <i>et.al.</i> [18]	54	63.1	67.7	70.5	72.3	73.6
MKSR (Method 1)	58.34	66.81	70.83	74.02	76.1	77.8
MKSR (Method 2)	58.9	67.3	71.44	74.7	76.83	78.01

as the SIFT-ScSPM, except that the local SIFT descriptor was replaced by the self-similarity descriptor [47]. The size of the patch and the radius of the window were fixed at 5×5 and 40 respectively.

LBP-ScSPM: Another image descriptor was constructed using the ScSPM procedure, for Local Binary Pattern (LBP) [48] features extracted from overlapping patches in an image.

Gist: The images were resized to 128×128 and the gist descriptor was extracted from each image [49]. The kernel matrix was constructed based on Euclidean distance between the features.

PHOG: We extracted the PHOG descriptor from each image using the procedure described in [50], by fixing the number of pyramid levels at 2. We used the Euclidean distance, and the χ^2 distance between the features for *Method 1* and *Method 2* respectively.

Biologically inspired features (C2-SWP, C2-ML): Biologically inspired C2 features proposed in [51] and [52] aim to mimic the simple and complex features in the human visual system. We extracted C2-SWP and C2-ML features, and used the Euclidean distance for both cases.

Geometric blur: For each image, we randomly sampled 400 edge pixels and applied the geometric blur descriptor [53] to them. We used the distance function proposed in [41] with these descriptors. Note that, because of the form of the distance function, this feature cannot be used in a fixed point dictionary update scheme. Therefore, when using this feature with *Method 1*, we use the initial dictionary obtained in the input space and refrain from updating it in the kernel space.

In order to initialize the algorithms for obtaining MKSRs, the parameters in the descriptors and the distance functions were tuned independently. The criteria for tuning is that the resulting sparse codes with the base kernels individually achieved their best performances in classification using a linear SVM. When optimizing dictionaries and computing MKSRs, the ensemble kernel weights $\{\beta_r\}_{r=1}^R$ were empirically tuned, again to ensure high classification accuracies with a test set. To achieve this, we repeated the algorithms by the randomly split-

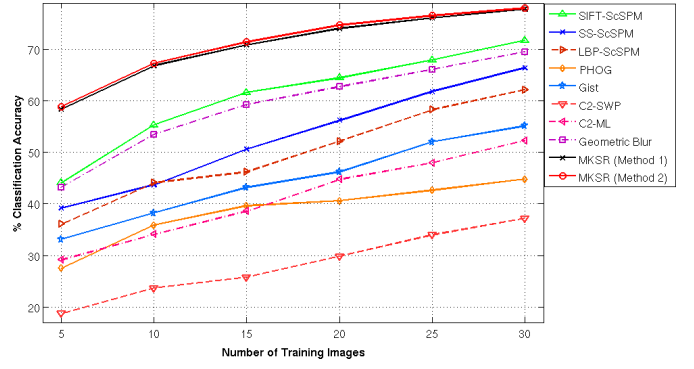


Fig. 3. Classification performance obtained by using each base kernel on the Caltech-101 dataset, in comparison to multiple kernel sparse representations.

ting the data samples into train and test sets, and determined the weights using cross-validation. We used the MATLAB interface of LIBLINEAR, a fast implementation of linear SVM for all our simulations [59]. The performance metric used is the percentage classification accuracy.

In addition to object recognition, spectral clustering can be performed using the graph created from the kernel sparse codes. Using the coefficient matrix \mathbf{B} , we constructed the non-negative graph weight matrix $\mathbf{W} = |\mathbf{B}^T \mathbf{B}|$. The normalized weight matrix is given as $\mathbf{L} = \mathbf{D}^{-1/2} \mathbf{W} \mathbf{D}^{-1/2}$, where \mathbf{D} is a diagonal matrix with the i^{th} diagonal element containing the sum of the all the elements in the i^{th} row or column of \mathbf{W} . Assuming that there are C clusters, the eigenvectors corresponding to the C largest eigenvalues are stacked in the matrix $\mathbf{U} = [\mathbf{u}_1 \mathbf{u}_2 \dots \mathbf{u}_C]$. The rows of the matrix are then clustered to obtain the cluster labels. The standard metrics - clustering accuracy and normalized mutual information (NMI), are used to quantify the clustering performance [29].

B. Simulation Results

1) *Caltech-101:* The Caltech-101 dataset [60] consists of 9144 images belonging to 101 object categories and an additional class of background images. The number of images in each category varies roughly between 40 and 800. We resized all images to be no larger than 300×300 with the aspect ratio preserved. Following the common evaluation procedure, we trained the classifiers on 5, 10, 15, 20, 25 and 30 images per class and evaluated the performance by testing on the rest.

For the proposed *Method 1*, we obtained separate K-Means dictionaries (2048 atoms) for each descriptor set in the original input space. Following this, we computed the ensemble kernel matrices using the expressions in (13), (14) and (15) respectively. The sparsity penalty λ for computing the kernel sparse codes was fixed at 0.3. We also performed kernel dictionary learning using the proposed *Method 2* described in Section V-D fixing the number of levels at 16 and the number of atoms per level at 128, resulting in a total of 2048 atoms.

The quantitative results of the proposed multiple kernel sparse representations frameworks are presented in Table III. The recognition rates presented are averaged over 10 iterations with train and test datasets chosen at random. As it can be observed, the proposed approaches achieve higher classification

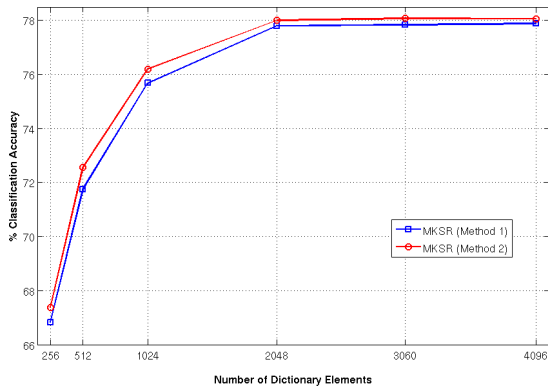


Fig. 4. Classification accuracies of the proposed MKSR algorithms on the Caltech-101 dataset using dictionaries of different sizes.

rates in comparison to other sparse coding based approaches. In order to demonstrate the importance of fusing the multiple kernels, we performed object recognition by considering each base kernel separately. In each case, we tuned the kernel parameter to yield the best recognition performance with sparse codes obtained using KMLD. Figure 3 illustrates the classification accuracies obtained, with each base kernel, for different number of training images per class. For comparison, we show the accuracies obtained for the two proposed multiple kernel methods as well. The improvement in recognition by using multiple kernels is apparent in all cases. For example, when $N_{train} = 15$ the SIFT-ScSPM and the geometric blur descriptors provide the best accuracies of 61.65% and 59.29% respectively, while the C2-SWP descriptor achieves a very low recognition rate of 25.8%. However, when all the kernels are combined using the proposed methods we achieve improvements of 9.18% and 9.79% in the mean recognition performance, when compared to using just SIFT-ScSPM descriptors.

Finally, we demonstrate the effect of dictionary size on the classification performance for both the proposed methods. We varied the number of dictionary elements between 256 and 4096 and repeated the simulations with 30 training samples per class, using 10 different random train and test sets in each case. Figure 4 plots the mean percentage classification accuracy for different dictionary sizes. We observed that beyond $K = 2048$, the classification rate does not improve significantly with the size of the dictionary.

We evaluated the clustering performance of the kernel sparse coding-based graphs using a benchmark subset of the Caltech-101 dataset (20 classes) [32]. Similar to the object recognition simulations, we constructed graphs using kernel sparse codes of the individual features for comparison. The dictionary size was fixed at 2048 for all cases. Table IV shows the clustering accuracy and normalized mutual information obtained using the different graphs. In each case, both the average and maximum values obtained over 50 trials are reported. As it can be observed, the ensemble features perform significantly better than the individual features.

2) *Caltech-256*: The Caltech-256 dataset [54] contains 30,607 images in 256 categories and its variability makes it extremely challenging in comparison to the Caltech-101

TABLE IV
COMPARISON OF THE CLUSTERING PERFORMANCE OBTAINED USING GRAPHS, CONSTRUCTED FROM THE KERNEL SPARSE CODES, ON A SUBSET OF CALTECH-101.

Feature	% Accuracy		NMI	
	Ave.	Max.	Ave.	Max.
SIFT-ScSPM	60.6	61.3	0.63	0.66
SS-ScSPM	52.8	54.31	0.52	0.59
PHOG	45.4	46.1	0.47	0.51
Gist	44.6	46.9	0.41	0.48
C2-SWP	38.31	39.6	0.33	0.36
C2-ML	31.8	33.1	0.29	0.35
GB	49.6	50.2	0.5	0.53
MKSR (Method 1)	69.8	74.7	0.74	0.81
MKSR (Method 2)	70.4	75.13	0.79	0.83

TABLE V
COMPARISON OF THE CLASSIFICATION ACCURACIES ON THE CALTECH-256 DATASET.

Method	# Training samples per class			
	15	30	45	60
Gemert <i>et.al.</i> [58]	-	27.17	-	-
Griffin <i>et.al.</i> [54]	28.3	34.1	-	-
Yang <i>et.al.</i> [17]	27.73	34.02	37.46	40.14
Guo <i>et.al.</i> [37]	29.77	35.67	38.61	40.3
Wang <i>et.al.</i> [19]	34.46	41.19	45.31	47.68
MKSR (Method 1)	38.72	45.12	48.65	50.27
MKSR (Method 2)	39.22	45.61	49.02	50.51

dataset. The experimental setup is similar to the previous section, and we evaluated the recognition performance with the number of training images fixed at 15, 30, 45, and 60 images per class respectively. The number of dictionary elements were fixed at 4096 for both the algorithms. Table V shows the recognition rates, and similar to the previous case, the proposed algorithms outperform other baseline methods.

VII. CONCLUSIONS

In this paper, we introduced the paradigm of multiple kernel sparse models that provide the ability to perform sparse learning in a unified space obtained using multiple kernels. The proposed approaches are of importance in complex vision problems, wherein they allow us to effectively characterize the image data, by fusing a large set of visual features. By learning sparse models in the multiple kernel domain, we generated compact representations for the images and achieved discrimination in the codes by exploiting the interactions between the multiple descriptors. Two different approaches for obtaining MKSR and optimizing the dictionaries for the feature space were developed. The proposed algorithms were comprehensively evaluated in supervised object recognition and unsupervised clustering tasks and the simulation results demonstrate the effectiveness of the proposed MKSR algorithms. The proposed models can be further extended, with appropriate constraints, to employ the resulting sparse codes in other frameworks such as semi-supervised learning.

REFERENCES

- [1] B. A. Olshausen and D. J. Field, "Sparse coding with an overcomplete basis set: A strategy employed by V1?" *Vision Research*, vol. 37, no. 23, pp. 3311–3325, December 1997.

- [2] M. Aharon, M. Elad, and A. Bruckstein, "K-SVD: An algorithm for designing overcomplete dictionaries for sparse representation," *IEEE Trans. on Signal Processing*, vol. 54, no. 11, pp. 4311–4322, 2006.
- [3] M. Elad and M. Aharon, "Image denoising via sparse and redundant representations over learned dictionaries," *Image Processing, IEEE Trans. on*, vol. 15, no. 12, pp. 3736–3745, 2006.
- [4] D. Donoho, "Compressed sensing," *Information Theory, IEEE Trans. on*, vol. 52, no. 4, pp. 1289–1306, 2006.
- [5] M. Zibulevsky and B. Pearlmutter, "Blind source separation by sparse decomposition in a signal dictionary," *Neural computation*, vol. 13, no. 4, pp. 863–882, 2001.
- [6] J. Wright, A. Yang, A. Ganesh, S. S. Sastry, and Y. Ma, "Robust face recognition via sparse representation," *IEEE Trans. on PAMI*, vol. 31, no. 2, pp. 210–227, 2009.
- [7] K. Yu and T. Zhang, "Nonlinear learning using local coordinate coding," in *Proc. of NIPS*, 2009.
- [8] S. S. Chen, D. L. Donoho, and M. A. Saunders, "Atomic decomposition by basis pursuit," *SIAM Review*, vol. 43, no. 1, pp. 129–159, 2001.
- [9] H. Lee, A. Battle, R. Raina, and A. Ng, "Efficient sparse coding algorithms," in *Proc. of NIPS*, 2007.
- [10] B. Efron, T. Hastie, L. Johnstone, and R. Tibshirani, "Least angle regression," *Annals of Statistics*, vol. 32, pp. 407–499, 2004.
- [11] J. A. Tropp, "Greed is good: Algorithmic results for sparse approximation," *IEEE Trans. on Information Theory*, vol. 50, no. 10, pp. 2231–2242, October 2004.
- [12] R. Rubinstein, A. Bruckstein, and M. Elad, "Dictionaries for sparse representation modeling," *Proc. of the IEEE*, vol. 98, no. 6, pp. 1045–1057, 2010.
- [13] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Optimality and stability of the k-hyperline clustering algorithm," *Pattern Recognition Letters*, vol. 32, no. 9, pp. 1299–1304, 2011.
- [14] R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Self-taught learning: Transfer learning from unlabeled data," in *Proc. of ICML*, 2007.
- [15] K. Grauman and T. Darrell, "The pyramid match kernel: Discriminative classification with sets of image features," in *Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on*, vol. 2. IEEE, 2005, pp. 1458–1465.
- [16] S. Lazebnik, C. Schmid and J. Ponce, "Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories," in *Proc. of IEEE CVPR*, New York, Jun. 2006.
- [17] J. Yang, K. Yu, Y. Gong and T. Huang, "Linear spatial pyramid matching using sparse coding for image classification," in *Proc. of IEEE CVPR*, Miami, Jun. 2009.
- [18] Z. Jiang, Z. Lin, and L. Davis, "Learning a discriminative dictionary for sparse coding via label consistent K-SVD," in *IEEE CVPR*, 2011.
- [19] J. Wang, J. Yang, F. Lv, T. Huang, and Y. Gong, "Locality-constrained linear coding for image classification," in *IEEE CVPR*, 2010.
- [20] J. Mairal, F. Bach, J. Ponce, G. Sapiro, and A. Zisserman, "Supervised dictionary learning," in *Proc. of NIPS*, 2009.
- [21] D. Bradley and J. Bagnell, "Differential sparse coding," in *Proc. of NIPS*, 2008.
- [22] Q. Zhang and B. Li, "Discriminative K-SVD for dictionary learning in face recognition," in *IEEE CVPR*, 2010.
- [23] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Sparse representations for pattern classification using learned dictionaries," in *Research and Development in Intelligent Systems XXV*, 2009, pp. 33–45.
- [24] S. Bengio, F. Pereira, Y. Singer, and D. Strelow, "Group sparse coding," in *Proc. of NIPS*, 2009.
- [25] J. J. Thiagarajan, K. N. Ramamurthy, P. Sattigeri, and A. Spanias, "Supervised local sparse coding of sub-image features for image retrieval," in *IEEE ICIP*, 2012.
- [26] S. Gao, I. Tsang, L. Chia, and P. Zhao, "Local features are not lonely laplacian sparse coding for image classification," in *IEEE CVPR*, 2010.
- [27] K. N. Ramamurthy, J. Thiagarajan, P. Sattigeri, and A. Spanias, "Learning dictionaries with graph embedding constraints," in *Proc. of Asilomar SSC*, 2012.
- [28] M. Zheng, J. Bu, C. Chen, C. Wang, L. Zhang, G. Qiu, and D. Cai, "Graph regularized sparse coding for image representation," *Image Processing, IEEE Trans. on*, vol. 20, no. 5, pp. 1327–1336, 2011.
- [29] B. C. B. Cheng, J. Y. J. Yang, S. Y. S. Yan, Y. F. Y. Fu, and T. S. Huang, "Learning with ℓ_1 -graph for image analysis," *IEEE Trans. on Image Processing*, vol. 19, no. 4, pp. 858–866, 2010.
- [30] I. Ramirez, P. Sprechmann, and G. Sapiro, "Classification and clustering via dictionary learning with structured incoherence and shared features," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 3501–3508.
- [31] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge, UK: Cambridge University Press, 2004.
- [32] Y. Lin, T. Liu, and C. Fuh, "Multiple kernel learning for dimensionality reduction," *IEEE Trans. on PAMI*, vol. 33, no. 6, pp. 1147–1160, Jun. 2011.
- [33] S. Sonnenburg, G. Rätsch, C. Schäfer, and B. Schölkopf, "Large scale multiple kernel learning," *The Journal of Machine Learning Research*, vol. 7, pp. 1531–1565, 2006.
- [34] M. Gonen and E. Alpaydin, "Multiple kernel learning algorithms," *Journal of Machine Learning Research*, vol. 12, pp. 2211–2268, 2011.
- [35] S. Gou, Q. Li, and X. Zhang, "A new dictionary learning method for kernel matching pursuit," in *FSKD*, 2006, pp. 776–779.
- [36] P. Vincent and Y. Bengio, "Kernel Matching Pursuit," *Machine Learning*, vol. 48, no. 1-3, pp. 165–187, 2002.
- [37] S. Gao, I. Tsang, and L. Chia, "Kernel sparse representation for image classification and face recognition," in *ECCV*, ser. Lecture Notes in Computer Science, 2010.
- [38] H. Nguyen, V. M. Patel, N. M. Nasrabadi, and R. Chellappa, "Kernel dictionary learning," in *IEEE ICASSP*, 2012.
- [39] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," *Computer Vision and Image Understanding*, vol. 106, no. 1, pp. 59–70, 2007.
- [40] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," 2007.
- [41] H. Zhang, A. C. Berg, M. Maire, and J. Malik, "Svm-knn: Discriminative nearest neighbor classification for visual category recognition," in *Proc. of IEEE CVPR*, 2006, pp. 2126–2136.
- [42] J. J. Thiagarajan, K. N. Ramamurthy and A. Spanias, "Learning stable multilevel dictionaries for sparse representation of images," *IEEE Trans. on Neural Networks and Learning Systems (Under review)*, 2013.
- [43] J. J. Thiagarajan, K. N. Ramamurthy, and A. Spanias, "Multilevel dictionary learning for sparse representation of images," in *IEEE DSP/SPE Workshop*, Sedona, Jan. 2011, pp. 271–276.
- [44] J. Thiagarajan, K. Ramamurthy, and A. Spanias, "Mixing matrix estimation using discriminative clustering for blind source separation," *Digital Signal Processing*, 2012.
- [45] Z. He, A. Cichocki, Y. Li, S. Xie, and S. Sanei, "K-hyperline clustering learning for sparse component analysis," *Signal Processing*, vol. 89, pp. 1011–1022, 2009.
- [46] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal on Computer Vision*, vol. 60, pp. 91–110, November 2004.
- [47] E. Shechtman and M. Irani, "Matching local self-similarities across images and ideos," in *Proc. of IEEE CVPR*, June 2007.
- [48] M. Pietikainen, A. Hadid, G. Zhao, and T. Ahonen, *Computer vision using Local Binary Patterns*, ser. Computational Imaging and Vision. Springer, 2011.
- [49] A. Oliva and A. Torralba, "Modeling the shape of the scene: A holistic representation of the spatial envelope," *International Journal on Computer Vision*, vol. 42, pp. 145–175, May 2001.
- [50] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns," in *Proc. of IEEE ICCV*, 2007.
- [51] J. Mutch and D. G. Lowe, "Multiclass object recognition with sparse, localized features," in *Proc. of IEEE CVPR*, vol. 1, 2006, pp. 11–18.
- [52] T. Serre, L. Wolf, and T. Poggio, "Object recognition with features inspired by visual cortex," in *Proc. of IEEE CVPR*, 2005, pp. 994–1000.
- [53] A. C. Berg and J. Malik, "Geometric blur for template matching," in *Proc. of IEEE CVPR*, vol. 1, Los Alamitos, CA, USA, 2001.
- [54] G. Griffin, A. Holub, and P. Perona, "Caltech-256 object category dataset," California Institute of Technology, Tech. Rep. 7694, 2007. [Online]. Available: <http://authors.library.caltech.edu/7694>
- [55] P. Jain, B. Kulis, and K. Grauman, "Fast image search for learned metrics," in *Proc. of IEEE CVPR*, Jun. 2008, pp. 1–8.
- [56] O. Boiman, E. Shechtman, and M. Irani, "In defense of nearest-neighbor based image classification," in *Proc. of IEEE CVPR*, Aug. 2008, pp. 1–8.
- [57] D. Pham and S. Venkatesh, "Joint learning and dictionary construction for pattern recognition," in *Proc. of IEEE CVPR*, vol. 0, 2008, pp. 1–8.
- [58] J. C. Gemert, J. Geusebroek, C. Veenman, and A. W. Smeulders, "Kernel codebooks for scene categorization," in *Proc. of ECCV*, 2008, pp. 696–709.
- [59] R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, "Liblinear: A library for large linear classification," *The Journal of Machine Learning Research*, vol. 9, pp. 1871–1874, 2008.
- [60] L. Fei-Fei, R. Fergus, and P. Perona, "Learning generative visual models from few training examples: An incremental bayesian approach tested on 101 object categories," in *IEEE CVPR*, Jun. 2004.